Tampereen yliopisto

Johannes Pirhonen

GSM-VERKKOA KÄYTTÄVÄ DATALÄHETIN

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Joulukuu 2019

TIIVISTELMÄ

Johannes Pirhonen: GSM-verkkoa käyttävä datalähetin
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaatin tutkinto-ohjelma, Sähkötekniikka
Joulukuu 2019

Tämän kandidaatintyön tarkoituksena oli suunnitella ja rakentaa GSM-lähetin mehiläispesän etävalvontaan. Idea työlle tuli paimiolaiselta mehiläistarhauksen ammattilaiselta. Laitteen tarkoituksena ja työn motivaationa oli myös oman mehiläisharrastuksen hunajasadon mittaaminen ja sadon valvominen. Mehiläispesän painon seurannalla voidaan vähentää turhia tarhakäyntejä, mikä säästää aikaa ja kustannuksia. Toisaalta ajatuksena oli myös edistää mehiläistarhauksen digitalisaatiota ja automatisointia Suomessa.

Suunniteltava laite kytketään lisäosana vaakaan, joka mittaa kentällä olevan mehiläispesän painoa. Lähetin lukee vaa'an RS-232-väylän kautta painotiedot ja lähettää ne edelleen tekstiviestillä ja sähköpostilla käyttäjälle. Lähetin mittaa ulkolämpötilaa, vaa'an akun jännitettä, sekä omaa käyttöjännitettään. Näin käyttäjä saa tiedon tarhalla vallitsevasta lämpötilasta, sekä mahdollisesta tarpeesta vaihtaa laitteen akut. Koko järjestelmän virrankulutus on tarkoitus saada niin alhaiseksi, että laitteisto pystyy toimimaan itsenäisesti ilman huoltoja vähintään satokauden, eli noin 4 kuukautta. Vaikka laitetta tullaan käyttämään ulkona vaihtelevissa olosuhteissa, sääsuojaukseen ei käytetty tässä työssä resursseja, sillä lähetin sijoitetaan vaa'an sääsuojatun käyttöliittymän kotelon sisään.

Laitteesta valmistui ensimmäisen prototyypin jälkeen vielä päivitetty versio, johon liittimien paikkoja, komponenttiarvoja ja komponenttikoteloita säädettiin. Tätä versiota valmistettiin käsityönä 8 kappaletta, joista 6 myytiin pilottiasiakkaille. Ohjelmistoa säädettiin vähitellen vuoden ajan testiasiakkaiden palautteen mukaan, jonka jälkeen laitteet ovat toimineet jopa 2 vuotta itsenäisesti. Voidaan todeta, että kaikki työn tekniset tavoitteet saavutettiin, mutta ajallisesta suunnitelmasta täytyi joustaa merkittävästi. Myös aineettomat tavoitteet, kuten järjestelmän toimittaminen ja arvon luominen asiakkaille toteutui. Lisäksi työ kartutti merkittävästi kokemusta elektroniikkalaitteen suunnitteluun, toteutukseen ja projektiluontoiseen työhön liittyen.

Avainsanat: AVR, Atmel, GSM, sim800L, embedded, C

ALKUSANAT

Tämän kandidaatintyön tekemiseen käytetty aika ja resurssit ylittivät moninkertaisesti suositukset, sillä kokonainen elektroniikkalaitteen suunnitteluprosessi ja etenkin ohjelmiston kirjoittaminen osoittautui suureksi haasteeksi. Prototyyppien myynnin yhteydessä lupasin testiasiakkaille takuun, joten laitteet oli saatava toimimaan toivotulla tavalla ajallisista suunnitelmista huolimatta. Iteratiivinen virheidenkorjausprosessi jatkui yli vuoden. Lopulta ohjelmisto saatiin riittävän toimintavarmaksi.

Suuri kiitos kuuluu datalähetintyön ensisijaiselle toimeksiantajalle ja testaajalle, Ismo Kajanderille. Hän sai ohjelmaversioita nopeasti toimimattomaan tilaan kerta toisensa jälkeen käyttäessään prototyyppilaitteita kentällä. Kiitos myös ohjaajalle Erja Sipilälle kannustuksesta, kärsivällisyydestä ja positiivisuudesta.

Erityiskiitos vanhemmilleni suuresta henkisestä tuesta, kannustuksesta ja kärsivällisyydestä työn eri vaiheissa. Kiitos isälle aikataulupaineen ylläpitämisestä ja laatikon ulkopuolelta ajatelluista kommenteista: "Onko maajohto kiinni?". Tämä ratkaisi erään sarjaliikenneongelman myöhään illalla.

Lopuksi kiitos kumppanilleni Arjalle valtavasta kannustuksesta ja kärsivällisyydestä. Loppusuoralla kannustus ja akateemiset vinkit antoivat merkittävästi tukea kirjallisen raportin valmistumiselle.

Tampereella, 20. joulukuuta 2019

Johannes Pirhonen

SISÄLLYSLUETTELO

KUVALUETTELO

OHJELMA- JA ALGORITMILUETTELO

LYHENTEET JA MERKINNÄT

ADC	Analogia-digitaalimuunnin (engl. Analog to Digital Converter)
ASCII	Kirjoitusmerkkien koodausstandardi (engl. American Standard Code for Information Interchange)
AT-komennot	Tekstipohjainen modeemien välinen komentokieli (engl. Attention/-Hayes Command Set)
ATmega328P	8-bittinen AVR mikrokontrolleri
AVR	Atmelin (nyk. Microchip) valmistama, Harvard arkkitehtuuriin perustuva 8-bittinen mikrokontrolleriperhe
BJT	Bipolaaritransistori (engl. Bipolar Junction Transistor)
EMC	Sähkömagneettinen yhteensopivuus (engl. Electromagnetic Compatibility)
ESR	Sisäinen sarjaresistanssi (engl. Equivalent Series Resistance)
GPIO	Mikrokontrollerin sisäämeno-/ulostulosignaalipinni (engl. General-purpose Input/Output)
GSM	Maailmanlaajuinen matkapuhelinjärjestelmä (engl. Global System for Mobile Communications)
HTTP	Verkkolaitteiden tiedonsiirtoprotokolla (engl. Hypertext Transfer Protocol)
I/O-rajapinta	Mikrokontrollerin signaalin sisäänmeno-/ulostulorajapinta (engl. Input/Output)
IDE	Ohjelmiston kirjoittamiseen ja kehittämiseen tarkoitettu tietokoneohjelma (engl. Integrated Development Environment)
ISP	Laitteeseen asennetun mikrokontrollerin tai logiikkapiirin ohjelmoinnin mahdollistava menetelmä (engl. In-system Programming)
LCD	Nestekidenäyttö (engl. Liquid-crystal Display)
MOSFET	Eristehilatransistori (engl. Metal-oxide-semiconductor Field-effect Transistor)
NiMH	Nikkelimetallihydridiakku (engl. Nickel Metal Hydride)
NPN-transistori	Bipolaaritransistori, jonka kollektori ja emitteri ovat n-tyypin puolijohdetta.
RS-232	Kahden tietokoneen välistä tietoliikennettä määrittelevä standardi (engl. Recommended Standard 232)

TQFP	Mikropiirin kotelointistandardi (engl. Thin Quad Flat Package)
TTM	Tuotteen tai palvelun kehittämisessä käytetty aika idean syntymishetkestä myyntikelpoiseksi tuotteeksi asti (engl. Time to Market)
UART	Tietokoneen tai mikrokontrollerin ja niiden oheislaitteiden välille kehitetty dataliikennerajapinta (engl. Universal Asynchronous Receiver Transmitter)
URC	GSM-standardin mukainen AT-protokollan automaattinen ilmoituskoodi (engl. Unsolicited Result Code)
USB	Tietokoneen oheislaitteille ja tiedonsiirrolle kehitetty sarjaväyläarkkitehtuuri (engl. Universal Serial Bus)

1 JOHDANTO

Nyky-yhteiskunnassa yhä useammin digitaalinen laite helpottaa tai jopa suorittaa kokonaan tehtäviä, joita vain ihmiset tekivät ennen. Etäohjattujen laitteiden määrä teollisuudessa ja kuluttajamarkkinoilla kasvaa nopeasti, ja yhä useampi elektroninen laite tai kone on osa jatkuvasti laajenevaa, niin sanottua Esineiden Internetiä [1, 2]. Työn ja laskennan automatisoinnilla ja edullisesti saatavilla olevien laitteiden avulla on tarkoitus ensisijaisesti säästää aikaa [3]. Mikäli apulaitteita käytetään oikein ja ne integroidaan järkevästi työprosessiin, ajansäästö voi olla hyvinkin merkittävä [4]. Ajallinen säästö on monesti verrannollinen taloudelliseen säästöön, jota jokainen voittoa tavoitteleva yritys ja yrittäjä toivoo.

Opintojeni ohessa harjoittamani mehiläistarhaus on alana melko vähän digitalisoitu, ja tarkoitukseen suunniteltuja apulaitteita on kotimaan markkinoilla vielä niukasti. Kollegan pyynnöstä aloin kehittää laitetta, joka olisi suunniteltu alusta alkaen käyttöympäristön haasteet huomioiden. Suurimpia haasteita ovat virrankulutus ja toimintavarmuus. Laitteen huoltovälin on oltava mahdollisimman pitkä, sillä tarhaajan on vaikea suorittaa huoltotoimenpiteitä suojavarusteissa. Siksi laitteen akkujen on riitettävä vähintään satokauden ajan, jotta tarhoilla käydessä voidaan keskittyä mehiläisten hoitoon.

Tässä työssä esitellään rakentamani yksinkertaisen elektroniikkalaitteen tuotesuunnitteluprosessi, jonka tuloksena on GSM-verkossa (engl. Global System for Mobile Communications) toimiva, akkukäyttöinen datalähetin. Laitteen suunnittelun kolmeksi päätavoitteeksi linjattiin vähävirtaisuus, toimintavarmuus ja edullisuus. Rakennettava laite integroidaan mehiläispesän vaakaan, jolloin saadaan tulokseksi etäohjattava mikrokontrollerijärjestelmä. Laitteiston tehtävä on lähettää mehiläistarhaajalle tiedot mehiläispesän painosta, ulkolämpötilasta ja laitteiston tilasta kerran vuorokaudessa. Tarhaajan ja lähettimen kommunikaatiorajapintana toimii yksinkertainen tekstiviestipohjainen protokolla. Tämän avulla valitaan lähettimen lähetystavat, lähetettävät tiedot sekä säädetään asetuksia.

Hunajantuottajana yksi merkittävä menoerä yrityksen kirjanpidossa on autokulut. Autolla on liikuttava joskus huomattavia matkoja yksinkertaistenkin asioiden vuoksi, sillä sopivimmat maastot ovat usein kymmenien kilometrien päässä taajamista ja asuinalueista. Mehiläisyhdyskunta tarvitsee satokaudella paljon erilaisia hoitotoimenpiteitä, joista yksi tärkeimmistä on yhdyskunnan sikiöinti- ja hunajaosastojen tilan hallinta. Mehiläiset alkavat tuntea olonsa ahtaaksi, jos yhdyskunnan pesälaatikot pääsevät täyttymään liikaa. Tällöin kuningatar ja yli puolet kaikista yhdyskunnan mehiläisistä jättävät pesän. Tarhaajan näkökulmasta tämä ei ole toivottua, sillä tällöin kyseisen yhdyskunnan hunajantuo-

tanto katkeaa hetkellisesti. Myös kallisarvoinen kuningatar ja suuri osa työläisistä menetetään. Kesällä suotuisat sademäärät ja voimakkaat hellejaksot voivat vauhdittaa mehiläisten hunajantuotantoa lyhyessä ajassa erittäin paljon.

Mehiläisyhdyskunnan painon muutoksesta ajan suhteen voidaan arvioida lisätilan tarve ja esimerkiksi päätellä, onko pesän hylännyt merkittävä määrä mehiläisiä lyhyen ajan sisällä. Mahdollisimman reaaliaikainen informaatio yhdyskunnan tilasta voi auttaa ratkaisevasti oikea-aikaisen huoltotarpeen arvioinnissa. Tässä työssä esitelty datalähetin pyrkii vastaamaan kysyntään suomalaiselle laitteelle, joka on vähävirtainen ja luotettava.

Luvussa kaksi, kandidaatintyön teoriaosuudessa, esitellään aiheen keskeisiä käsitteitä. Kolmannessa luvussa käsitellään lähettimen systeemitason spesifikaatioita, mistä siirrytään esittelemään lähettimen elektroniikka- ja piirilevysuunnittelua. Neljännessä luvussa käsitellään laitteelle kirjoitettua ohjelmakoodia, jaotteluna pääohjelma ja toiminnalliset lohkot. Lopuksi vedetään yhteen toteutuksessa kohdattuja ongelmia ja niiden ratkaisuja sekä pohditaan projektin mahdollisuuksia jatkokehityksen kannalta. Dokumentin liiteosiossa on esillä lähettimen kytkentäkaavio, piirilevysuunnitteluun liittyvät dokumentit, pääohjelman lohkokaavio ja ohjelmakoodia.

2 LÄHETTIMEN KESKEISET AIHEALUEET

Tässä luvussa käsitellään työn ymmärtämisen kannalta keskeisimpiä käsitteitä ja aihepiirejä. Luku on jaettu laitteiston kannalta oleellisiin lohkoihin. Rakennettavan laitteiston tehtävä on lähettää mehiläistarhaajalle tiedot mehiläispesän painosta, ulkolämpötilasta ja laitteiston tilasta kerran vuorokaudessa. Tarhaajan ja lähettimen kommunikaatiorajapinnaksi kirjoitettiin tekstiviestipohjainen komentoprotokolla. Tekstiviestikomentojen avulla valitaan lähettimen lähetystavat, lähetettävät tiedot, kysytään GSM-liittymän saldo sekä säädetään asetuksia.

2.1 Mikrokontrolleri

Mikrokontrolleri on sulautetun järjestelmän ydinkomponentti, joka vastaa yksin tai yhdessä muiden kontrollereiden kanssa systeemin oheislaitteiden toiminnan ohjaamisesta. Usein käytetään myös termiä mikro-ohjain. Mikrokontrolleri koostuu yhden komponenttikotelon sisään rakennetuista toimintalohkoista [5]. Kontrollerin keskusyksikön ympärille on rakennettu muistilohkot, keskeytyslogiikka ja I/O-rajapinnat (sisäänmeno-/ulostulorajapinnat, engl. Input/Output) [5, 6], joiden toimintaan perehtyminen ei ole tämän työn ymmärtämisen kannalta kriittistä. Tässä työssä käytetään vähävirtaista Microchipin (ent. Atmel) valmistamaa ATmega328P-mikrokontrolleria jalallisena pintaliitoskotelovaihtoehtona (TQFP, engl. Thin Quad Flat Package).

Kontrolleri on ohjelmoitava mikrotietokone, joka lukee sisäänmenoiksi asetettuihin pinneihin tulevia signaaleita ja muuttaa ulostuloiksi asetettujen pinnien tilaa kontrollerin ohjelmassa määritetyllä tavalla. Näin kontrollerilla voidaan mitata ympäröivän maailman fysikaalisia suureita erilaisten anturien avulla. Anturit muuttavat fysikaalisen suureen sähköiseksi informaatioksi. Usein tämä informaatio on jännite-ero jonkin komponentin, kuten resistiivisen elementin napojen välillä. Mikrokontrolleri on digitaalinen laite, joten luettavat analogiset signaalit on muunnettava digitaalisiksi ennen kuin niitä pystytään käsittelemään. Monissa mikrokontrollereissa on sisäänrakennettuna analogia-digitaalimuunninlohko (ADC, engl. Analog to Digital Converter), eli analogia-digitaalimuunnin, joka mahdollistaa luettavan signaalinäytteen tallentamisen digitaaliseksi tiedoksi kontrollerin muistiin. Antureilta saatua informaatiota voidaan käyttää erilaisten toimintojen ohjaamiseen ja säätämiseen. Mitattua tietoa voidaan tarvittaessa lähettää edelleen muille laitteille johtimia pitkin tai langattomasti. [5, 7, 8]

2.2 GSM-moduuli ja AT-komennot

Suunniteltavan datalähettimen päätehtävä on tiedon lähettäminen langattomasti käyttäjälle. Laitteen käyttöympäristö on usein kaukana taajamista, joten yhteystekniikaksi valikoitui perinteinen puhelinverkko. Verkkoon kytkeytymiseen yksinkertaisin vaihtoehto on valmis GSM-moduuli. GSM-verkko tunnetaan myös toisen sukupolven puhelinverkkona, joka kattaa nykyisellään aukottomasti lähes koko suomen [9]. [10]

Mikrokontrollerin ja GSM-moduulin kommunikaatiorajapintana toimii GSM-standardin mukainen komentoprotokolla (AT-komennot, engl. Attention). Mikrokontrolleri ja GSM-moduuli keskustelevat asynkronisen sarjaväylän (UART, engl. Universal Asynchronous Receiver Transmitter) kautta käyttäen GSM-standardin mukaisia AT-komentoja [11]. AT-komentoprotokolla on modeemien väliseen viestintään suunniteltu kokonaisvaltainen tekstipohjainen kommunikaatiomenetelmä, jolla voidaan ohjata GSM-moduulin toimintoja [12]. Moduuli vastaa komentoihin ilmoittaakseen komennon vastaanottamisesta ja mahdollisista komennon suorittamisen lisätiedoista.

2.3 Ohelmiston kehitysympäristö ja valmiit funktiokirjastot

Lähettimen ytimenä toimiva ATmega328P mikrokontrolleri voidaan ohjelmoida käyttäen puhdasta C-kieltä, assembleria, tai näiden yhdistelmää [13]. Kyseisen kontrollerin ympärille on olemassa myös suuren suosion saavuttanut italialainen avoimen lähdekoodin Arduino ekosysteemi [14], jossa lähdekoodin ohjelmointikielenä on laajalti C++. Arduino ekosysteemiin kuuluu useita kehitysalustoja, lisälaitteita sekä kattava ohjelmistokirjasto erilaisten sulautettujen elektroniikkaprojektien pohjaksi. Sama ekosysteemi tarjoaa myös yksinkertaisen kehitysympäristön (IDE, engl. Integrated Development Environment). Arduino IDE sisältää pelkistetyn koodieditorin, johon on sisäänrakennettu Arduinon ohjelmistokirjastot ja helpot toiminnot ohjelman kääntämiseksi ja lataamiseksi kontrollerille. [13, 14]

Arduinon tarjoaman editorin yksinkertaisuuden vuoksi tässä työssä käytettiin ohjelman kirjoittamiseen ulkoista Notepad++ tekstieditoria, jonka jäsennys- ja hahmotusominaisuudet ovat Arduinon ympäristöä kattavammat. Ohjelman kääntäminen ja lataaminen kontrollerille on tehty Arduino IDE:n kautta, jotta voitiin hyödyntää helposti etenkin Arduinon ohjelmallisen sarjayhteyden SoftwareSerial-kirjastoa.

Valmiiden ohjelmistokirjastojen etu on helppokäyttöisyys ja ihmisläheisyys. Kirjastot sisältävät matalampien abstraktiotasojen koodin ohjelmoijalta piilossa. Toisaalta nämä voi osoittautua haasteeksi toimimattoman ohjelman virheiden löytämisessä. Korkean tason ohjelmointikielet ja niiden ohjelmistokirjastot ovat usein suunnattu prosessoreille, joissa muisti ja laskentateho eivät ole niin merkittäviä rajoitteita, kuin mikrokontrollerijärjestelmissä [15]. Vaikka nykyaikaisten ohjelmakääntäjien optimointiominaisuudet ovat hyvin kehittyneitä, ohjelmistokirjastojen yleismallisia funktioita käyttämällä ei pystytä aina hyö-

dyntämään kontrollerin ohjelmamuistia täysin optimaalisesti. Tällöin ohjelmoijalla ei välttämättä ole täyttä kontrollia suoritettavista komennoista kellojakson tarkkuudella. Mikäli käytettävässä kontrollerissa on riittävästi laskentatehoa ja muistia, korkean tason ohjelmointi voi olla edullisin vaihtoehto prototyyppivaiheessa nopean testauksen ja markkinoilletuontiajan (TTM, engl. Time to Market) kannalta. [13, 15, 16]

3 LÄHETTIMEN ELEKTRONIIKKASUUNNITTELU

Piirikaavio ja -levy suunniteltiin Mentor Graphics:n PADS-ohjelmistolla. Liitteessä A on esitetty viimeisin versio lähettimen kytkennästä. Lähettimen spesifikaatioita ja liityntöjä määrittelee suurilta osin isäntälaitteeksi valittu teollisuuskäyttöinen vaaka. Toisaalta koko järjestelmän virrankulutus on saatava mahdollisimman alhaiseksi ottaen huomioon GSM-moduulin määrittelemä käyttöjännitealue ja virranantokyky. Vähävirtaisuutta on pyritty huomioimaan suunnittelun jokaisessa vaiheessa.

Kuvassa 3.1 nähdään valittu vaakalaite, joka koostuu rungon ja siihen kiinnitetyn punnitusanturin lisäksi nestekidenäytöstä (LCD, engl. Liquid Crystal Display) ja käyttöliittymäpaneelista, jonka kotelon sisään lähetin asennetaan. Käyttöliittymäpaneelin sisällä on RS-232 (engl. Recommended Standard 232) sarjaliikennestandardin mukainen tiedonsiirtoväylä, jonka kautta Atmel AVR-tuoteperheen 8-bittisen ATmega328P-mikrokontrolleri lukee painotiedot. Kontrollerin digitaalinen ohjauspinni PC4 on kytketty optoerottimeen, jonka kautta kontrollerilla voidaan ajaa suoraan vaa'an virtakatkaisijaa.

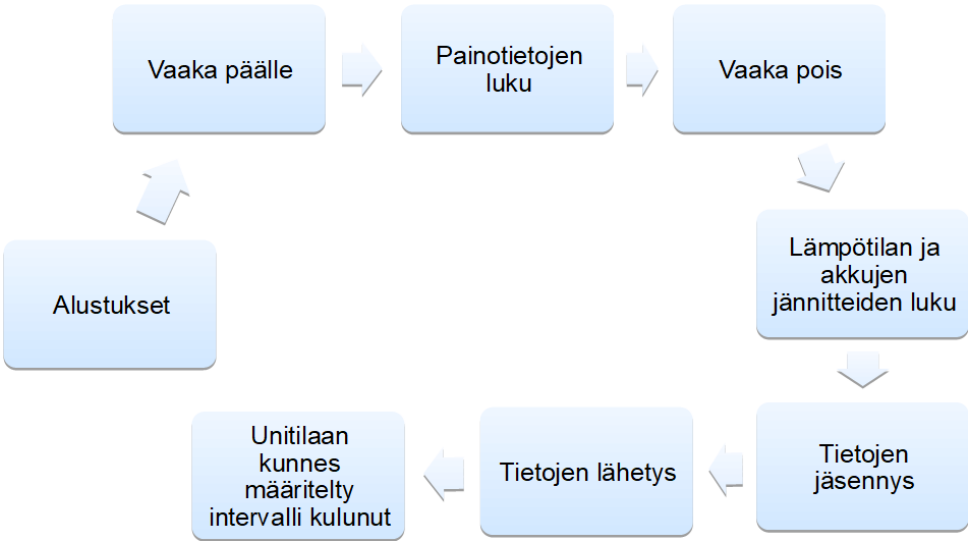
Kuvassa 3.2 nähdään lähettimen suunniteltu toiminta systeemitasolla. Alustuttuaan lähetin lukee asetusviestit sim-kortilta, käynnistää vaa'an, jäsentää sarjaportin kautta saadut painotiedot ja sammuttaa vaa'an. Tämän jälkeen kontrolleri mittaa isäntälaitteen lyijyakun jännitteen, oman käyttöjännitteensä, vallitsevan ulkolämpötilan ja lähettää kerätyt tiedot käyttäjälle joko sähköpostitse tai tekstiviestillä GSM-moduulin avulla. Käyttäjä voi asettaa tekstiviestikomennolla tekstiviesti- ja sähköpostilähetykset toisistaan riippumatta päälle tai pois. Lopuksi mikrokontrolleri siirtyy unitilaan, kunnes herää vahtikoira-ajastimen avulla vuorokauden päästä suorittamaan saman toimenpideketjun.

Kontrollerin ADC-lohkon mittaaman jännitteen on oltava pienempi kuin ADC:n käyttämä referenssijännite. Vaa'an lyijuakun nimellisjännite on noin 6 V ja ADC-referenssinä käytetään kontrollerin noin 3,6 V käyttöjännitettä. ADC-lohkon sisäänmenopinnin jännite on siis säädettävä yksinkertaisella jännitteenjaolla. Jännitteenjakovastukset on valittu suuriksi, jotta jännitteenmittauspiiri kuluttaisi mahdollisimman vähän virtaa vaa'an lyijyakulta. Kontrollerin datalehdeltä [17] nähdään kuitenkin, että ADC-pinniin kytketyn piirin ulostuloimpedanssin on oltava alle 10 kΩ, jotta ADC:n sample-and-hold -kondensaattori saa tarpeeksi virtaa latautuakseen ja mittaustulos olisi luotettava. Tästä syystä jännitteenjakopiirin alemman vastuksen rinnalle on kytketty kondensaattori, josta ADC-lohko saa lisävirtaa mittausta suoritettaessa.

Vaa'an lähettämien painotietojen lukemiseksi laitteen kytkennässä on NPN-tyypin bipo-

laaritransistorilla (BJT, engl. Bipolar Junction Transistor) ja vastuksilla toteutettu yksinkertainen signaalitasojen muunnin, jota käsitellään kohdassa 3.4. Seuraavissa aliluvuissa käsitellään lähettimen eri toimintalohkojen suunnittelua ja toteutusta.

Kuva 3.1. *Vaakana toimii teollisuuskäyttöinen laite, jonka käyttöpaneelin sisään lähetin asennetaan.*



Kuva 3.2. *Lähettimen toiminta systeemitasolla lohkokaavioesityksenä.*

3.1 ATmega-kontrolleri ja suodatuskomponentit

Työn aloittamisaikana ATmega328P oli ainoa tuttu ohjelmoitava kontrolleri, joten yksinkertaisuuden vuoksi tämä valikoitui rakennettavan lähettimen ydinkomponentiksi. Lisäksi ATmega328P-kontrolleria käytettiin Mikrokontrollerit-kurssilla, jolla valmistettiin Arduino Uno R3 -kehitysalustan kopio. Tässä työssä rakennettu lähetin pohjautuu vahvasti edellä mainittuun kehitysalustaan. Käyttämällä ATmega328P-kontrolleria, myös Arduino ohjelmointiympäristö ja sen kirjastot ovat suoraan käytettävissä.

Arduinon mallikehitysalustan piirikaaviossa [18] nähtävät tehonsäätökomponentit on jätetty kokonaan pois lähettimen piirikaaviosta. Lisäksi kytkennän yksinkertaistamiseksi USB-rajapinta (engl. Universal Serial Bus) on karsittu lähettimen piirikaaviosta, jolloin kontrollerin ohjelmointi tapahtuu erillisellä ohjelmointilaitteella ohjelmointiväylän (ISP, engl. In-system Programming) kautta. Lähettimen piirikaaviosta nähdään, että ATmega328P:n käyttöjännitteiden suodatuskondensaattorit ja kaksi ferriittiä ovat identtisiä Arduino mallikehitysalustan [18] kanssa.

ATmega-kontrollerissa on vain yksi UART-sarjaliikenneportti, mutta sarjayhteyttä käyttäviä oheislaitteita on kaksi: GSM-moduuli ja vaaka. Jälkimmäisen sarjayhteys toteutettiin ohjelmallisesti, sillä kontrollerin ja GSM-moduulin välinen kommunikaatio arvioitiin järjestelmän kannalta kriittisemmäksi. Toteutukseen valittu Arduino SoftwareSerial -kirjasto osoittautui toimivaksi ratkaisuksi. Kirjastoa jouduttiin kuitenkin muokkaamaan, koska mikrokontrollerin käyntitaajuudeksi valittiin SoftwareSerial-kirjastolle ja koko Arduino-ympäristölle epätyypillinen 3,6864 MHz.

3.2 Kontrollerin kellosignaali

Suunnitteluvaiheen alussa tarkoituksena oli kalibroida kontrollerin sisäinen oskillaattori ja käyttää sitä lähettimen kellosignaalina. Sisäinen oskillaattori vie vähemmän virtaa kuin ulkoinen oskillaattori, ja lisäksi säästyttäisiin ylimääräisiltä komponenteilta. Kuitenkin aiheeseen perehtymisen myötä päädyttiin käyttämään erillistä kvartsikidettä kellosignaalina. Mikrokontrollerin sisäinen oskillaattori on yleisesti virhealttiimpi ja epätarkempi kuin ulkoinen kide [17]. Lisäksi sisäisen oskillaattorin tarkkuus on vahvasti lämpötilariippuvainen, mikä voi olla ongelma lämpötilan vaihdellessa kymmeniä asteita ulko-olosuhteissa. Ulkoinen kvartsikide on lähes immuuni lämpotilamuutoksille, värähtelytaajuus on tarkasti tiedossa eikä kalibrointia tarvita.

Laitetta tahdistava kvartsikide valittiin käyttöjännitteen ja sarjaliikenteen luotettavuuden mukaan. ATmega328P –kontrollerille on määritelty luotettavan toiminnan kannalta turvalliset käyttöjännitealueet eri käyntitaajuuksille. Kontrollerin datalehdeltä [17, s. 312] voidaan nähdä, kuinka käyttöjännitteen tarve on suoraan verrannollinen mikrokontrolleripiiriä tahdistavan kellotaajuuden kanssa. Kontrollerin liian korkea kellotaajuus suhteessa käyttöjännitteeseen voi aiheuttaa laitteen toiminnan epävakautta.

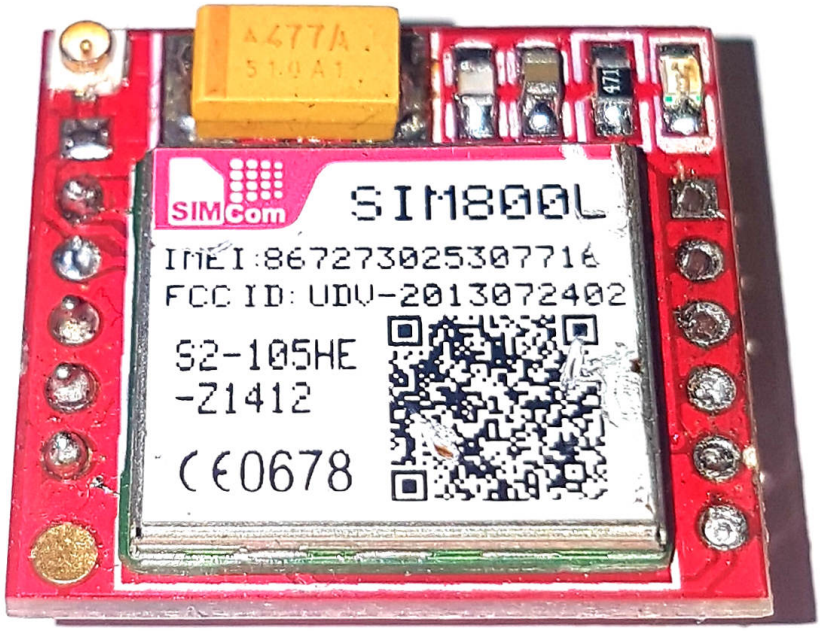
Lähettimen on tarkoitus olla syvimmässä unitilassa suurimman osan ajasta. Tällöin ulkoinen kide pysäytetään eikä kiteen virrankulutus ole merkittävä koko käyttöaikaan suhteutettuna. Optimaaliseksi kelloksi valikoitui 3,6864 MHz:n kvartsikide, sillä kyseinen taajuus mahdollistaa kaikilla yleisimmillä sarjaliikennenopeuksilla 0 % virheen kontrollerin sarjaväylän bittien tahdistuksessa. Tämä on suotavaa, sillä lähettimen kriittisimpiä tehtäviä on kommunikoida GSM-moduulin kanssa luotettavasti sarjaliikenneväylän kautta. Lisäksi valittu taajuus on linjassa lähettimen käyttöjännitteen kanssa.

3.3 Tehonsyöttö

Lähettimeen valittu GSM-moduuli on laitteen vaativin komponentti virrankäytön kannalta. Sim800L-moduuli tarvitsee hetkellisesti virtaa jopa kaksi ampeeria. Huippuvirtoja tarvitaan millisekuntien ajan moduulin rekisteröityessä lähimpään mobiiliverkon tukiasemaan ja lähettäessä tekstiviestiä mobiiliverkon yli. Laitteen tehonlähteen on siis pystyttävä tarjoamaan tarpeeksi virtaa ilman jännitteen putoamista. GSM-moduuli vaatii 3,3–4,5 voltin jännitteen toimiakseen luotettavasti. Jännitteen laskiessa alle 3,4 voltin moduulin sisäänrakennettu alijännitesuoja liipaistuu ja moduuli alustuu ennen tiedonsiirtotoimenpiteen onnistumista. Moduulilevyllä kuvassa 3.3 on virtapiikkejä varten käyttöjännitepinnin rinnalle juotettu suuri 470 mikrofaradin tantaalikondensaattori. Tämä avustaa kytkettyä energialähdettä virtapiikkien aikana vapauttaen nopeasti energiaa.

Lähettimen kytkennän suunnittelussa on huomioitu virtapiikit valitsemalla energianlähteeksi Panasonicin Eneloop NiMH-paristot (engl. Nickel Metal Hydride). Valinnan perustana oli riittävä dokumentaatio, ladattavuus, matalien lämpötilojen sieto ja etenkin paristojen pieni sisäinen sarjaresistanssi (ESR, engl. Equivalent Series Resistance) [19]. Paristojen matala ESR-arvo mahdollistaa pariston sisäisten jännitehäviöiden pysymisen maltillisena virtapiikkien aikana. Jännitelähteen sisäisten häviöiden ollessa liian suuria, jännite laskee alle GSM-moduulin vaatiman minimijännitteen ja moduuli alustuu.

Eneloop-paristojen nimellisjännite on 1,2 V. Tämän ansioista on mahdollista käyttää kolmea paristoa sarjassa suoraan lähettimen jännitelähteenä. Näin vältytään erillisten tehoa kuluttavien lineaariregulaattorien tai häiriöisten hakkurien käytöltä lähettimen tehonsyötössä. Tehonsäätöä ei tässä tilanteessa välttämättä tarvita, sillä jännitteen kannalta kriittisiä toimintalohkoja on vain kaksi, lähettimen kontrolleri ja GSM-moduuli. Lähettimen kontrolleria tahdistava kvartsikide on valittu myös käyttöjännite huomioiden. Tällöin kontrollerin on mahdollista toimia luotettavasti käyttöjännitealueella, joka on valittu GSM-moduulin mukaan. Yksinkertaisella tehonsyötöllä vähennetään laitteen kytkennän kompleksisuutta merkittävästi, valmistuskustannukset pienenevät ja mahdollisten vikaantuvien komponenttien määrä vähenee. Kun jännitelähteenä käytetään suoraan paristoja ilman tehonsäätökomponenttien vuotovirtoja, saavutetaan myös hieman alhaisempi virrankulutus.

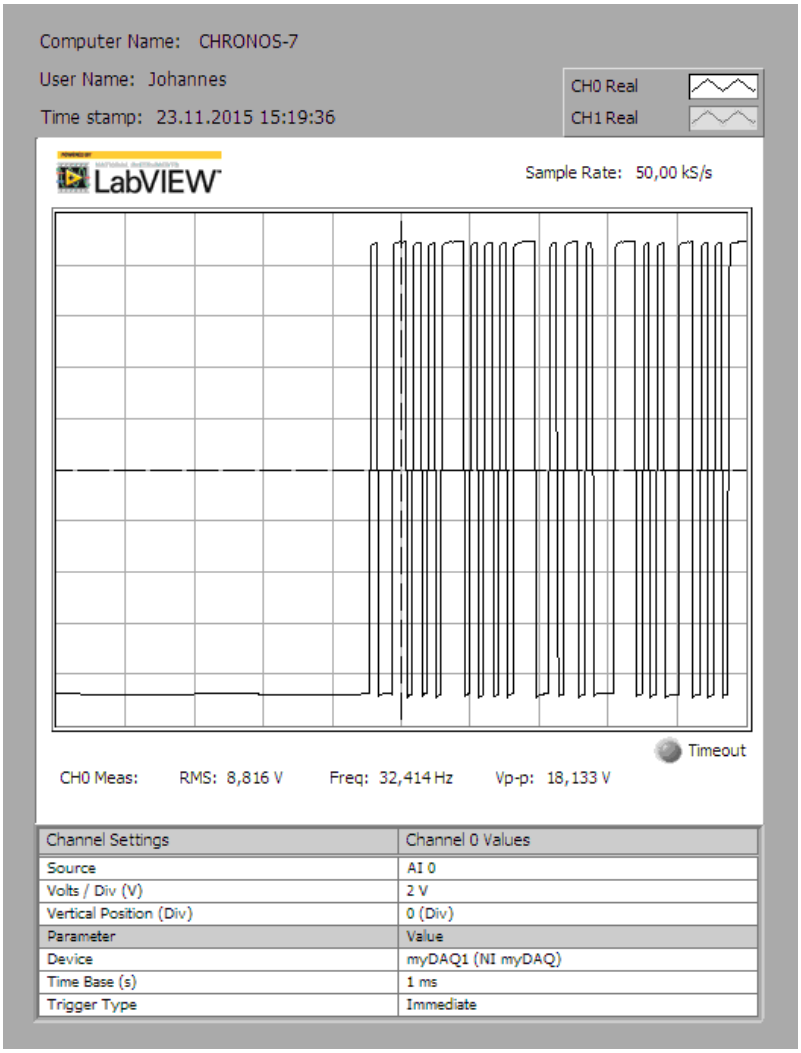


Kuva 3.3. *Sim800L GSM-moduuli adapterilevyllä.*

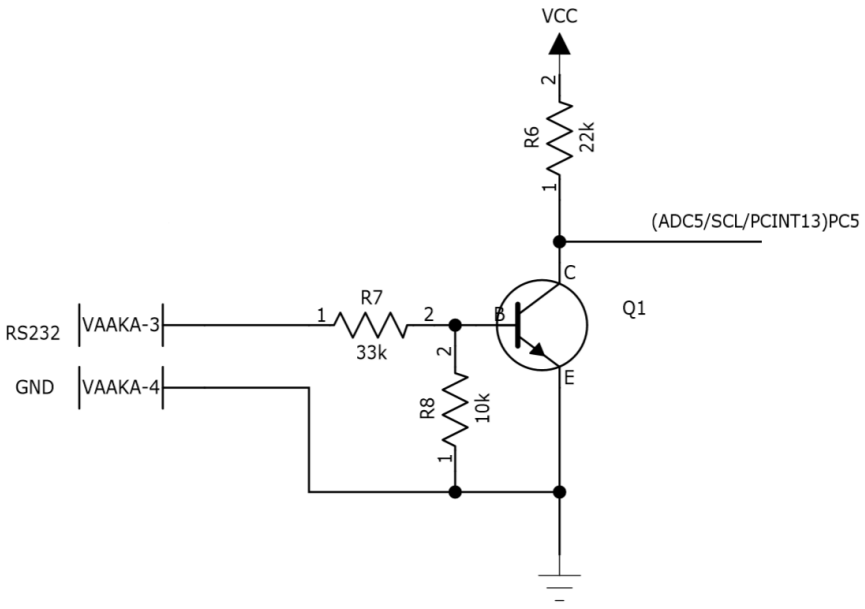
3.4 RS-232 muunnin

Vaakalaitteen käyttöpaneelissa on RS-232 standardin mukainen sarjaliikenneväylä. Painotiedot saadaan lähettimen käsiteltäväksi luotettavasti sarjaväylän kautta. RS-232 esiteltiin jo vuonna 1960 määrittelemään tietokoneiden ja niiden oheislaitteiden välistä kommunikaatiota. Standardi määrittelee laitteen ulostulosignaalitasot enintään -15 V ja +15 V. Negatiivinen jännite on looginen 1, positiivinen on looginen 0 ja lähtöjännitteen muutosnopeus pitää olla vähintään 30 V/μs. [20]

Lähettimen mikrokontrollerijärjestelmä toimii eri jännitealueella ja käänteisellä logiikalla. Yksinkertainen invertoiva BJT-kytkentä mahdollistaa sarjaliikenteen muuntamisen sopivaksi ATmega-kontrollerille. Muunninkytkennän biasointivastusten mitoittamiseksi tarkastettiin oskilloskoopilla vaa'an sarjaväylän jännitetasot, jotka nähdään kuvassa 3.4 olevan -9 V ja +9 V. Kuvassa 3.5 nähdään muunninkytkentä, jossa vastukset R7 ja R8 biasoivat NPN-transistorin. RS-232-signaalin ollessa -9 V, transistorin V_{BE}-kanava on estosuuntaan biasoitu. Tällöin transistori ei johda ja SoftwareSerialin sisäänmenopinnin PC5 jännite on ylhäällä ylösvetovastuksen R6 myötä. RS-232 ulostulon ollessa +9 V, V_{BE}-kanava on myötäsuuntaan biasoitu ja transistori johtavassa tilassa. Tällöin PC5 pinnissä näkyy spesifikaation [21] mukaan saturaatiotilassa olevan transistorin V_{CE}-jännite 0,05 V, minkä ATmega32P tulkitsee loogisena arvona nolla [17, s. 322].



Kuva 3.4. *Vaa'an sarjaliikenneväylän signaalin oskilloskooppikuvaaja.*



Kuva 3.5. *Invertoiva NPN-transistorikytkentä RS-232 muuntimena.*

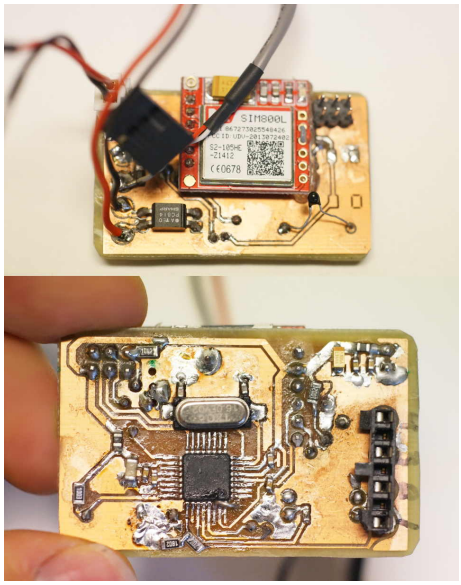
3.5 GSM-moduulin MOSFET-kytkin

Sim800-sarjan GSM-moduulia on saatavana yleisesti adapterilevylle valmiiksi ladottuna, missä tarvittavat signaalit on tuotu piikkirimoille. Adapterilevyllä on valmiina suodatuskondensaattorit, indikaattoriledi ja ledin etuvastus. Adapterilevyissä on kuitenkin virtakatkaisinpinni sidottu käyttöjännitteeseen, joten moduulia ei ole mahdollista saada pysymään täysin pois päältä katkaisematta sen käyttöjännitettä. Liitteessä A esitetyssä piirikaaviossa nähdään kuinka GSM-moduulin käyttöjännitteen katkaisemiseen käytetään eristehilatransistoria (MOSFET, engl. Metal Oxide Semiconductor Field Effect Transistor) yksinkertaisena kytkimenä. Huomataan myös, että saman kytkimen takana on vallitsevan lämpötilan mittauksen piirilohko. Katkaisemalla myös lämpötilanmittauspiiri unijakson ajaksi, virrankulutusta voidaan pienentää edelleen.

Avaustyypin N-MOSFET:n hilajännitettä ohjataan mikrokontrollerin digitaalisella signaalilla. Transistorin hila on biasoitu niin, että hilan jännite on mahdollisimman korkea ohjauspinnin ollessa ylhäällä. Tällöin transistori toimii saturaatioalueellaan johtavassa tilassa. MOSFET:n erityispiirre on suuri sisäänmenoimpedanssi ja alhainen jännitehäviö lähteen ja nielun yli, jolloin GSM-moduulille jää mahdollisimman suuri osa käyttöjännitteestä. [22]

3.6 Piirilevyn suunnittelu

Kaksipuolisen piirilevyn valmistus ja ladonta tehtiin Tampereen Yliopiston Sähkötekniikan yksikön oppilaslaboratorioissa. Levyn valmistuksessa käytettiin menetelmää, jossa levyn fotoresistiivinen lakka valotetaan ultraviolettivalolla ja piirilevykuvio etsataan suolahappoliuoksella. Liitteessä B on esitetty piirilevykuvioinnin valmistuksessa käytetyt valotusmaskit.



Kuva 3.6. *Lähettimen ensimmäinen prototyyppi*

4 OHJELMISTON TOTEUTUS

Ohjelman suunnitteluvaiheessa käytettiin Mikrokontrollerit-kurssilla rakentamaani Arduino UNO R3 -kehitysalustan kloonia. Kuvassa 3.6 nähtävän prototyypin valmistuttua ohjelmointi ja testaus jatkui filmipurkin sisään Arduino Nano -kehitysalustasta tehdyllä ISP-ohjelmointilaitteella.

Ohjelmointiympäristönä käytettiin Arduino IDE -kehitysympäristöä ja avr-gcc kääntäjää. Arduino-kehitysympäristö on suunnattu aloitteleville harrastajille ja ohjelmointiympäristö sisältää paljon valmiita funktioita ja ohjelmakirjastoja ohjelmoinnin helpottamiseksi. Tällöin kuitenkin voi jäädä epäselväksi, mitä kontrolleri todellisuudessa tekee bittitasolla kullakin hetkellä. Tarkoituksena oli yksinkertaistaa joitakin mikrokontrollerin toimintojen ohjelmointia käyttämällä Arduino-kehitysympäristöä. Projektin edetessä lisää ominaisuuksia ja virhetilanteiden sietoa lisättäessä ilmeni, että Arduinon C++-tyylisiä funktioita oli käytettävä maltillisesti. Valmiit funktiot vievät paljon ohjelmamuistia, joten niitä päädyttiin käyttämään mahdollisimman vähän.

Muuttujien tyyppeihin oli kiinnitettävä huomiota, jottei niille varattaisi turhaan ylimääräistä tilaa. Esimerkiksi yksittäisille binääriarvoisille lipuille varattiin aluksi 8 bittiä muistia `uint8_t`-tyypin muuttujan muodossa. Ohjelmiston jatkojalostuksessa tällaisia muuttujia yhdistettiin toisiinsa niin, että yhdessä muuttujassa useampi bitti toimi loogisten tarkastelujen lippubittinä. Muuttujien operaatiot toteutettiin bittikohtaisesti bittimaskien avulla.

Ohelmakoodi on pyritty jakamaan loogisten kokonaisuuksien mukaan eri moduuleihin. Näin koodin hahmottaminen ja ylläpito helpottuu merkittävästi. Yksittäinen moduuli koostuu käytännössä 2 tiedostosta, moduulin julkisen rajapinnan määrittävästä `.h`-otsaketiedostosta ja varsinaisesta `.c`-lähdekooditiedostosta. Lähdekooditiedosto sisältää funktioiden määrittelyt ja toiminnallisuuden. Moduulia käyttävä toinen kooditiedosto viittaa moduulin otsaketiedostoon pystyäkseen käyttämään moduulin tarjoamia funktioita. Seuraavissa aliluvuissa esitellään ohjelmakoodin pääosia jaoteltuna pääohjelmaan ja erillisiin moduuleihin. Esitetyt ohjelmakoodin osat ovat erittäin pelkistettyjä tämän dokumentin rajallisen laajuuden vuoksi.

4.1 Pääohjelma

Pääohjelma voidaan jakaa kolmeen eri loogiseen kokonaisuuteen: alustuksiin, pääohjelman silmukkaan, ja paikallisten funktioiden määrittelyihin. Ensimmäisenä on suoritet-

tava kaikki viittaukset tarvittavien kirjastojen ja moduulien otsakkeisiin. Yleisesti on tapana viitata ensin kolmansien osapuolten tarjoamiin kirjastoihin ja sitten itse kirjoitettuihin. Otsakkeiden jälkeen määritetään vakiot `#define`-avainsanalla ja alustetaan globaalit muuttujat asianmukaisine tyyppeineen. Ohjelmassa 4.3 nähdään ote lähettimen pääohjelmakoodissa tehtävistä vakioiden ja muuttujien alustuksista, sekä sisäämeno-/ulostulosignaalipinnien (GPIO, engl. General-purpose Input/Output) ja UART-alustukset.

Pääohjelman silmukkaa suoritetaan kunnes laitteen akuista loppuu varaus tai ohjelma kaatuu. Silmukan toiminta lohkokaavioesityksenä on nähtävänä liitteessä C. Silmukan ensimmäinen lohko on lähettimen oman käyttöjännitteen, eli kolmen sarjaan kytketyn Eneloop-akun, jännitteen mittaus. Mittausta varten kirjoitettu funktio 4.1 käyttää referenssinä akkujen jännitettä, jota vastaan mitataan kontrollerin sisäinen 1.1 V jännitereferenssi. Todellinen lähdejännite saadaan laskettua manuaalisesti säädetyn `kalibrointi`-vakion ja mitatun jännitteen osamäärällä. [17, 23]

```
1  long mittaaVcc() {
2      long kalibrointi = 1090000L; // Kalibroitava sähkömittarin avulla
3      // Asetetaan referenssiksi AVcc ja mitattavaksi sisäinen 1.1V:
4      // ADMUX-rekisterin bitit 6(REFS0),1,2,3. (MUX1-3)
5      ADMUX |= 0b01001110;
6      _delay_ms(3); // Odotetaan käyttöjänniteen stabiloitumista
7      // Mitataan 1.1V referenssi AVcc:tä vasten
8      ADCSRA |= 0x04; // Alioitetaan mittaus, ADCSRA:n kuudes bitti
9      while (bit_is_set(ADCSRA,ADSC)); // Odotetaan kunnes valmis
10     uint8_t ala  = ADCL; // Kopioidaan mittaustuloksen alarekisteri
11     uint8_t yla = ADCH; // sitten ylärekisteri
12     long tulos = (yla<<8) | low; // Yhdistetään
13     tulos = kalibrointi / tulos; // lasketaan todellinen Vcc (mV)
14     return tulos; // Vcc millivoltteina
15 }
```

Ohjelma 4.1. *Eneloop-akkujen jännitteen mittaus [17, 23].*

Tämän jälkeen käynnistetään GSM-moduuli MOSFET-kytkimen avulla. On varmistettava, että moduuli rekisteröityy oikein operaattorin tukiasemaan. ATmega-kontrolleri kuuntelee GSM-moduulin UART:n lähetyspinnin signaalia, kunnes kaikki alustumisen ja tukiasemaan liittymisen varmistavat GSM-protokollan mukaiset ilmoituskoodit (URC, engl. unsolicited result code) ja operaattorikohtaiset rekisteröintiviestit on vastaanotettu. Tästä vaiheesta kontrolleri siirtyy suoraan unitilaan vuorokaudeksi ohittaen kaikki datankeruuvaiheet, jos GSM-moduulilta ei saada varmistusta onnistuneesta verkkoon rekisteröitymisestä.

Normaalissa toimintatapauksessa GSM-moduulilta vastaanotetaan kontrollerin odottamat URC-koodit ja teleoperaattorin viestit. Lähettimen asetuksia voidaan muuttaa tekstiviestitse, joten tässä vaiheessa kontrolleri odottaa operaattorilta mahdollisesti tulevia

käyttäjän viestejä. AT-komentoja käyttäen kontrolleri lukee GSM-moduulilta tekstiviestit ja mikäli oikeanmuotoisia komentoja löytyy, suoritetaan niitä vastaavat toiminnot. Käyttäjän komennot käsittelevä toiminta on toteutettu erillisenä ohjelmamoduulina, mitä käsitellään kohdassa 4.2.

Käyttäjän lähettämien asetuskomentojen käsittelyn jälkeen kontrolleri lukee vallitsevan lämpötilan termistorin avulla, kysyy verkon tukiasemalta kellonajan, jonka mukaan lasketaan tarvittava uniaika seuraavaan haluttuun lähetysajankohtaan asti. Kontrollerin uniaika on normaalitilanteessa noin yksi vuorokausi. Koska kontrollerilla ei ole erillistä reaaliaikakellolohkoa ja halutaan saavuttaa pieni virrankulutus, uni on toteutettu käyttäen syvintä unitilaa ja kontrollerin vahtikoira-ajastinlohkoa (WDT, engl. watchdog timer). Unijakson toteutusta käydään läpi aliluvussa 4.3.

Kontrolleri käynnistää vaa'an, lukee ja jäsentää painotiedon käyttäen muokattua SoftwareSerial-kirjastoa. Painotiedon luettuaan kontrolleri kytkee vaa'an pois päältä ja lähettää kerätyt tiedot käyttäjälle tekstiviestitse ja sähköpostitse, riippuen käyttäjän määrittelemistä asetuksista. Tässä dokumentissa käsitellään tekstiviestin lähetys. Ohjelmassa 4.2 nähdään yksinkertaistettu esimerkki painotiedon lähettämisestä. Laitteen varsinaisessa ohjelmassa on toteutettu laitteen nimien, painon, lämpötilan ja akkujännitteiden lisäksi toiminnallisuus GSM-verkon kentänvoimakkuuden ja liittymän saldon lähettämiseksi.

```
1  if(haluttu_lahetystapa & 0x01){ // Normaali tekstiviesti
2      sendAT("AT+CMGS=\"");
3      sendAT(puh_vastaanottaja);
4      sendAT("\"\r\n");
5      // Odotetaan GSM-moduulin kuittausta valmiudesta vastaanottaa
6      // teksti. Luetaan UART-väylää kunnes ei enää liikennettä.
7      while(!(UCSR0A & (1<<RXC0)));
8      while(!(UCSR0A & (1<<UDRE0)));
9      sendAT("Paino: ");
10     sendAT(painotiedot);
11 }
12 sendAT("\x1A\r\n"); // Hex 0x1A lähettää viestin
13 confirmAT(); // Varmistetaan, että GSM-moduuli vastasi "OK"
```

Ohjelma 4.2. *Esimerkki tekstiviestin lähettämisestä AT-komennoilla [24].*

Tässä vaiheessa GSM-moduulin avulla tehtävät toiminnot on suoritettu. Kontrolleri komentaa GSM-moduulin pois päältä UART-väylän kautta viestillä . Lisäksi kontrolleri vetää MOSFET-kytkimen ohjauspinnin alas, jotta GSM-moduulin lähdevirtapiiri ja lähettimen termistoripiiri ovat kokonaan auki. Käytetyn GSM-moduulin käynnistyspainikkeelle tarkoitettu powerkey-pinni on kyseisessä adapterilevyssä juotettu käyttöjännitteeseen, joten moduuli olisi aina päällä jos virtaa ei katkaista. Pääohjelman toimintosilmukan viimeinen kohta ennen univaihetta on itse unen pituuden laskeminen. Unijakson jälkeen suoritus jatkuu jälleen käyttöjännitteiden mittaamisella ja GSM-moduulin alustuksilla.

```
1  // Lähettimen tiedot
2  #define SARJANRO                "2"
3  #define OHJELMISTOVERSIO        "0.7.0"
4  #define KIDE                    "3,6864 Mhz"
5  // Termistorin vakioiden alustukset
6  #define TERMISTORIPINNI         A1
7  #define RINIMELLINEN            100000
8  #define NIMELLISLAMPO           25
9  #define NUMSAMPLES              100
10 #define BKERROIN                3942
11 #define SARJAVASTUS             100000
12 // Akun varaustilan mittaukseen vastusjako
13 #define R1                      1000.0
14 #define R2                      820.0
15
16 char laitteen_nimi[9] = "Vaaka 02";
17 char sposti_vastaanottaja[31] = "esimerkki@gmail.com";
18
19 // Lähetyksen asetukset
20 uint16_t haluttu_ilta_aika = 1260;
21 uint16_t haluttu_aamu_aika = 300;
22 uint8_t haluttu_lahetystapa = 0x02;
23 uint8_t kysy_kentta_saldo_debug = 0x00;
24 char puh_vastaanottaja[] = "1234567890";
25 uint16_t uni_minuuteissa = 1439;
26 // SoftwareSerial-olion alustus
27 SoftwareSerial swSerial(A5, A3, false); // RX, TX
28
29 void setup(){
30     DDRD |= (1 << PD4);          // MOSFETin-ohjauspinnni outputiksi
31     PORTD &= ~(1 << PD4);        // nollaksi -> N-MOSFET OFF
32     DDRC |= (1 << PC4);          // Vaa'an virtakytkin optoerotin
33     PORTC &= ~(1 << PC4);        // alustetaan 0 -> optoerotin OFF
34     DDRD |= (1 << PD1);          // UART TX = output
35     DDRD &= ~(1 << PD0);         // UART RX = input
36     // Haluttu baudinopeus: 9600 baud/s
37     // UBRR0 = (Fcpu / (16*baud rate))-1
38     UBRR0H = 0x00;
39     UBRR0L = 0x17;               // @ 3,6864 Mhz
40     // UART alustus
41     UCSR0B |= (1 << TXEN0) | (1 << RXEN0);
42 }
```

Ohjelma 4.3. *Muuttujien määrittely ja alustusfunktio.*

4.2 Moduulit

Ohjelman jakaminen erillisiksi moduulitiedostoiksi helpottaa ohjelman hahmottamista, hallintaa ja ylläpitoa. Ohjelmamoduuli on käytettävissä pääohjelmassa lisäämällä esimerkin 4.4 otsaketiedosto komennolla `#include poimi_komennot_viesteista.h` pääohjelman alkuun. Ohjelma 4.5 esittelee pelkistetyn version funktiosta, joka jäsentää ja toteuttaa käyttäjän lähettämät komennot vastaanotetuista tekstiviesteistä. Funktion parametreina on vastaanottajan puhelinnumero ja halutun lähetystapaosoittimen muistiosoite. Funktio lukee GSM-moduulin sarjaporttia kunnes kaikki tekstiviestisisältö on luettu simkortilta, eli vastaanotetaan peräkkäin merkit `"OK"` ja tavut `0x0D 0x0A`. Edellä mainitut tavut ovat ASCII-koodausstandardin (engl. American Standard Code for Information Interchange) mukaisesti kursorin palautus ja uusi rivi.

```
1  // Module: poimi_komennot_viesteista / file: poimi_komennot_viesteista.h
2  // Author: Johannes Pirhonen
3  #ifndef POIMI_KOMENNOT_VIESTEISTA_H
4  #define POIMI_KOMENNOT_VIESTEISTA_H
5
6  #include <Arduino.h>
7
8  bool poimi_komennot_viesteista( char puhelin[],
9                                  uint8_t &lahetystapa );
10
11 #endif // POIMI_KOMENNOT_VIESTEISTA_H
```

Ohjelma 4.4. *Komentojen jäsennysmoduulin otsaketiedosto.*

Funktion vikasietoisuutta on pyritty parantamaan lisäämällä Arduino-kirjaston `millis()`-ajastin. Kyseinen ajastinfunktio palauttaa ohjelman suorituksen aloittamisesta kuluneen ajan millisekunteina [23]. Tallentamalla aikaleima muuttujaan millä tahansa ajanhetkellä, ja vertaamalla tähän myöhemmin tallennettua aikaleimaa, voidaan laskea ohjelman eri osasuorituksiin kulunut aika. Tässä tapauksessa ajastimella luodaan ehto, jonka avulla poistutaan tekstiviestienlukusilmukasta halutun ajan kuluttua. Ehto mahdollistaa virheellisesti silmukkaan jääneen suorituksen aikakatkaisun ja ohjelman jatkamisen.

Komentojenjäsennysmoduulin lisäksi kokoelma lyhyitä avustavia funktioita on kerätty Apufunktiot-moduuliin. Apufunktioita ovat esimerkiksi UART-portin liikennettä ohjaavat funktiot ja erilaisten AT-komentojen vastauksia käsittelevät funktiot.

Kolmas pääohjelman käyttämä moduuli on aika- ja saldotietoja käsittelevä. Tässä moduulissa ajankyselyviestin vastauksena saatu operaattorin aikaleima luetaan ja jäsennetään myöhemmin unijakson laskentaan käytettäväksi. Lisäksi moduuli sisältää toteutuksen saldotietojen tilaamiseksi operaattorilta ja tietojen jäsentämiseksi. Käyttäjä voi myöhemmin pyytää tekstiviestikomennolla lähettimeltä saldotiedot osana normaalia viestiä.

```
1  // Module: poimi_komennot_viesteista / file: poimi_komennot_viesteista.cpp
2  // Poimii kellonajan Elisan matkapuhelinverkosta GSM-moduulin
3  // AT+CCLK? komennon jälkeen, ja palauttaa tunnit ja minuutit merkkijonoina
4  // Author:
5  // Johannes Pirhonen
6  #include "apufunktiot.h"
7  #include <Arduino.h>
8
9  bool poimi_komennot_viesteista( char puhelin[],
10                                  uint8_t &lahetystapa ){
11     char edellinen = 0x00;
12     unsigned long alkuMillis = millis();
13     char vastaanotettu;
14     int j = 0;
15     // Luetaan UART:ia kunnes kaikki viestit käyty läpi
16     while(j < 3){
17         unsigned long nytMillis = millis();
18         vastaanotettu = uart_getbyte();
19         // Vastaanottajan puhelinnumero komennolla
20         // #p1234567890
21         if ((edellinen == '#') & (vastaanotettu == 'p')){
22             for (int i=0; i<10; i++) {
23                 vastaanotettu = uart_getbyte();
24                 puhelin[i] = vastaanotettu;
25             }
26         }
27         // Tekstiviestilähetys päälle komennolla #T
28         else if ((edellinen == '#') &
29                 (vastaanotettu == 'T')){
30             lahetystapa |= 0x01;
31         }
32         // Viestit luettu loppuun kun saatu "OK", CR ja LF
33         else if ((edellinen == 'O') &
34                 (vastaanotettu == 'K')){
35             ++j;
36         }
37         else if ((j == 1) & (vastaanotettu == 0x0D)){
38             ++j;
39         }
40         else if ((j == 2) & (vastaanotettu == 0x0A)){
41             ++j;
42         }
43         else if ((nytMillis - alkuMillis) > 20000){
44             return false;
45         }
46         edellinen = vastaanotettu;
47     }
48     return true;
49 }
```

Ohjelma 4.5. *Komentojen jäsennys vastaanotetuista tekstiviesteistä.*

4.3 Syklisen unijakson toteutus

Laitteen vähävirtaisuustavoitteen vuoksi oli käytettävä mahdollisimman syvää kontrollerin unitilaa unijakson toteuttamiseen. Laite on aktiivisena noin minuutin vuorokaudesta, joten virrankulutuksen kannalta unijakson suunnittelu on erittäin merkittävää. Laitteeseen ei haluttu yksinkertaisuuden vuoksi lisäsä ulkoista reaaliaikakelloa, eikä GSM-moduuli ole enää unen aikana päällä, joten kontrollerin ei ole mahdollista saada ulkoista herätettä. Syvimmästä virransäästötilasta heräämiseen ainoa menetelmä ulkoisen herätteen lisäksi on kontrollerin sisäinen vahtikoira-ajastin.

ATmega328P-kontrollerin vahtikoira-ajastin käyttää lämpötilariippuvaista kalibroimatonta sisäistä oskillaattoria, joka on konfiguroitavissa maksimissaan noin 8 sekunnin mittaiseksi. Haluttu unijakson pituus on kuitenkin 24 tuntia, joten kontrolleri ohjelmoitiin heräämään aina vahtikoira-ajastimen umpeuduttua, noin 8 sekunnin välein. Näin voitiin käyttää syvintä unitilaa ilman ulkoista herätettä. Tämän ratkaisun heikkoutena kuitenkin on sisäisen oskillaattorin epätarkkuus ja lämpötilariippuvuus. Jokainen kontrolleriyksilö on hieman erilainen, joten jokaisen rakennetun lähettimen unilaskurin kaava on kalibroitava erikseen. Tässä projektissa unijakson ei tarvitse olla kuitenkaan kovin tarkka. Riittää, että käyttäjä saa lähettimeltä viestin suunnilleen asettamaansa aikaan. Riittäväksi tarkkuudeksi koettiin tässä tapauksessa ± 10 minuuttia. Epätarkkuus on huomioitu ohjelmakoodissa, jotta lähetin ei lähettäisi vuorokaudessa kahta viestiä herätessään hieman ennen asetettua lähetysaikaa. [17, 23]

Ohjelmassa 4.6 on esitetty unisyklien laskennan toteutus. Kontrolleri käynnistää vahtikoiran ja nollaa unisyklien laskurimuuttujan ja aloittaa silmukan, jota suoritetaan kunnes `uni_laskuri` saavuttaa `halutut_unisyklit`. Halutut syklit saadaan laskemalla halutun unen sekuntimäärän ja kalibroidun WDT-jakson osamäärä. Jokainen noin 8 sekunnin unijakso päättyy vahtikoiran laukaistessa keskeytyspalvelun, jossa kasvatetaan `volatile`-tyypin muuttujaa `uni_laskuri`.

```
1  uint16_t halutut_unisyklit = ((uni_minuuteissa*60)/8.20);
2  wdtOn();
3  uni_laskuri = 0;
4  // Kontrolleri unitilassa kunnes noin vuorokauden verran
5  // 8 sekunnin unijaksoja suoritettu.
6  while (uni_laskuri < halutut_unisyklit){
7      nukkumaan(); // Nukutaan noin 8 sekuntia
8  }
9  uni_laskuri = 0;
10 wdtOff();
11
12 void nukkumaan(){
13     set_sleep_mode(SLEEP_MODE_PWR_DOWN); // Asetetaan syvin unitila.
14     sleep_enable(); // Käynnistetään unitila.
```

```
15      sleep_mode(); // Kontrolleri valittuun unitilaan.
16      // Tässä herätään WDT:n laukaistessa keskeytyksen.
17      sleep_disable(); // Unitila pois päältä
18  }
19
20  ISR(WDT_vect){
21      uni_laskuri++; // Kasvatetaan unisyklien laskurimuuttujaa
22  }
```

Ohjelma 4.6. *Unijakson syklien laskenta WDT-keskeytyksen avulla [17].*

5 YHTEENVETO

Projektin tuloksena saatiin halutut toiminnot suorittava elektroniikkalaite, joka lopulta toimi koko vuoden ympäri virheettömästi. Suurin työmäärä oli ohjelmointivaiheessa. Koodin kirjoittaminen oli oppimisprosessi niin tyylillisesti kuin teknisesti. Ohjelman ensimmäinen kaikki halutut toiminnallisuudet sisältävä versio oli kirjoitettu yhteen tuhatriviseen tiedostoon. Virheiden löytäminen ja koodissa navigoiminen oli todella työlästä. Myöhemmin moduuleihin jako ja edistyneempään tekstieditoriin siirtyminen helpotti työtä.

Laitteen testijaksoissa ilmeni vielä lukuisia suunnitteluvirheitä. Elektroniikkasuunnittelun kannalta merkittävämmät heikkoudet olivat indikaattoriledien puuttuminen levyltä ja huonolaatuiset akkukotelot. Ledien puuttuminen vaikeutti ohjelmistokehitystä, sillä ei ollut mahdollista ilmaista ohjelman eri vaiheita suorituksen edetessä kontrollerilla. Käytettävistä akkukoteloista rikkoutuivat virtakatkaisijat helposti, joten ne päädyttiin oikosulkemaan ja kotelo kiinnittämään lähettimeen robustilla liittimellä. Levyn valmistuttua selvisi myös, että kontrollerilla on sisäinen lämpötilamittaus ominaisuus, joten ulkoista termistoria ei olisi tarvittu. Lisäksi haasteita loi edullisen GSM-moduulin puutteellinen dokumentaatio. Tietyissä moduulin valmistuserissä ei ollut tukea sähköpostin SSL-salaukselle, mikä esti yleisimpien sähköpostipalvelimien käytön tietyillä moduuleilla.

Ohjelmakoodissa ilmeni useita heikkouksia, joista merkittävin oli ilmeisesti SoftwareSerial-kirjaston ja `millis()`-funktion konflikti. SoftwareSerial-olion aloitusmetodia oli kutsuttava hallitusti vain sillä hetkellä kun vaa'alta luetaan painotiedot, ja lopetusmetodi kutsuttava heti kun mahdollista. Muussa tapauksessa SoftwareSerial-olio esti ohjelman toiminnan luotettavasti. Juurisyyn perusteellisempi selvittäminen ohitettiin.

Laitteeseen olisi tulevaisuudessa paljon kehitettävää. ATmega328P sisältää ainoastaan yhden todellisen UART-sarjaliitännän. Seuraavaan prototyyppiin voisi vaihtaa kontrollerin sovellukseen tarkoituksenmukaisemmaksi ATtiny841:ksi. Kyseisessä kontrollerissa on kaksi UART-porttia sekä vähemmän ylimääräisiä pinnejä. Tällöin on huomioitava ATtiny-kontrollerin pienempi 8 kilotavun muistitila ohjelmalle. Lisäksi laitteen transistorit ja optoerotin voitaisiin korvata huomattavasti pienemmillä pintaliitosvaihtoehdoilla. Vaikka testikäytössä ei ole kohdattu toimintaa häiritseviä EMC-ongelmia (engl. Electromagnetic Compatibility), laitteen häiriösuojausta olisi hyvä suunnitella perusteellisemmaksi. Lähettimen komentamisesta voisi tehdä reaaliaikaisemman heräämällä unitilasta GSM-verkkoon useammin vuorokaudessa, sillä akunkesto osoittautui hyväksi.

Toisaalta vaa'an oma lyijyakku on järjestelmän käyttöaikaa rajoittava tekijä, joten mah-

dollisuus kytkeä lähetin ja vaaka samaan energialähteeseen on tutkittava. Jos järjestelmästä haluttaisiin yksinkertaisempi, voitaisiin luopua kokonaan vaa'an käyttöpaneelista kytkemällä painoanturi ulkoisen ADC-muuntimen kautta suoraan lähettimen kontrolleriin. Tässä tapauksessa menetettäisiin kuitenkin mahdollisuus käyttää vaakaa myös LCD-paneelin kautta itsenäisesti kaikkine valmiine asetuksineen ja algoritmeineen.

Tekstiviesti- ja sähköpostilähetyksen lisäksi datan siirto suoraan nettisivulle HTTP-protokollan (engl. Hypertext Transfer Protocol) välityksellä mahdollistaisi datan graafisen esittämisen ja helpomman tilastoinnin. Lisäksi automaattiset ilmoitukset saldon loppumisesta, akkujen alhaisesta varauksesta tai painotietojen äkillisistä muutoksista voisivat parantaa merkittävästi laitteiston arvoa käyttäjän näkökulmasta.

Merkittävin opittu asia tämän projektin aikana, ohjelmointitaitojen kehittymisen lisäksi, oli elektroniikkalaitteen tuotekehitys prosessina. Erityisesti prototyyppivaiheen iteratiivisen luonteen oivallus. Jokainen toiminnallisuus on saatava testikäyttäjälle kokeiltavaksi mahdollisimman aikaisessa vaiheessa ja testitulosten perusteella kehitettävä toiminnallisuuksia edelleen. Liian valmiiden ja monimutkaisten ominaisuuksien rakentaminen tuotti myöhemmin hankaluuksia, sillä monesti viat ja kehitysideat selvisivät vasta ulkopuolisella testikäyttäjällä todellisessa käyttöympäristössä. Projektin myötä osoittautui, että loppukäyttäjä ja asiakas ovat projektin sidosryhminä ensiarvoisen tärkeitä koko projektin elinkaaren ajan.

LÄHTEET

[1] Evans, D. *How the Internet of Everything Will Change the World... for the Better. Cisco Blog*. 7. marraskuuta 2012. URL: https://blogs.cisco.com/digital/how-the-internet-of-everything-will-change-the-worldfor-the-better-infographic (viitattu 10. 05. 2019).

[2] Torchia, M. ja Shirer, M. *IDC Forecasts Worldwide Spending on the Internet of Things to Reach $745 Billion in 2019, Led by the Manufacturing, Consumer, Transportation, and Utilities Sectors*. 3. tammikuuta 2019. URL: https://www.idc.com/getdoc.jsp?containerId=prUS44596319 (viitattu 26. 05. 2019).

[3] Wong, E. M. C. A phone-based remote controller for home and office automation. *IEEE Transactions on Consumer Electronics* 40.1 (1994), pp. 28–34.

[4] Dineva, K. ja Atanasova, T. Model of Modular IoT-based Bee-Keeping System. *The 2017 European Simulation And Modelling Conference*. (Lisbon, Portugal). ResearchGate, lokakuu 2017, pp. 404–406. URL: https://www.researchgate.net/profile/Kristina_Dineva/publication/321860341_Model_of_Modular_IoT-based_Bee-Keeping_System/links/5a3578220f7e9b10d8451b25/Model-of-Modular-IoT-based-Bee-Keeping-System.pdf (viitattu 29. 06. 2019).

[5] Kuhnel, C. *AVR RISC Microcontroller Handbook*. 1. painos. Newnes, 1998, 312 p.

[6] Mitescu, M. ja Susnea, I. Resources of Microcontrollers. *Microcontrollers in Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–2. URL: https://doi.org/10.1007/3-540-28308-0_1 (viitattu 29. 05. 2019).

[7] Parab, J. S., Shinde, S. A., Shelake, V. G., Kamat, R. K. ja Naik, G. M. Introduction. *Practical Aspects of Embedded System Design using Microcontrollers*. Dordrecht: Springer Netherlands, 2008, pp. 1–18. URL: https://doi.org/10.1007/978-1-4020-8393-8_1 (viitattu 29. 05. 2019).

[8] Trevennor, A. *Practical AVR Microcontrollers: Games, Gadgets, and Home Automation with the Microcontroller Used in Arduino*. 1. painos. Apress, 2012, 401 p.

[9] *Elisa Oyj: Kuuluvuuskartta, Suomi*. URL: https://elisa.fi/kuuluvuus/ (viitattu 04. 08. 2019).

[10] Taferner, M. ja Bonek, E. *Wireless Internet Access over GSM and UMTS*. Springer, Berlin, Heidelberg, 2002, 224 p.

[11] European Telecommunications Standards Institute. *ETSI TS 127 007 Technical Specificaiton. Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; AT command set for User Equipment (UE)(3GPP TS 27.007 version 13.7.0 Release 13)*. Huhtikuu 2018. URL: https://www.etsi.org/deliver/etsi_ts/127000_127099/127007/13.07.00_60/ts_127007v130700p.pdf (viitattu 04. 08. 2019).

[12] A. Butterfield, G. E. Ngondi ja A. Kerr, toim. *A Dictionary of Computer Science*. 7. painos. Oxford University Press, 2016.

[13] Margush, T. S. *Some assembly required: assembly language programming with the AVR processor*. CRC Press, 2016, 624 p.

[14] Arduino SA. *What is Arduino*. 2019. URL: https://www.arduino.cc/en/Guide/Introduction (viitattu 11. 08. 2019).

[15] Sangiovanni-Vincentelli, A. ja Martin, G. Platform-based design and software design methodology for embedded systems. *IEEE Design & Test of Computers* 18.6 (2001), pp. 23–33.

[16] Kormanyos, C. *Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming*. 2nd 2015. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, 389 p.

[17] *ATmega328P Datasheet*. DS40002061A. Microchip. 2019, 622 p. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf (viitattu 03. 08. 2019).

[18] *Arduino Uno Rev3 Schematic*. Arduino SA. 2019. URL: https://content.arduino.cc/assets/UNO-TH_Rev3e_sch.pdf (viitattu 11. 08. 2019).

[19] *eneloop101.com. Panasonic Eneloop Test Results*. 2019. URL: https://eneloop101.com/batteries/eneloop-test-results/ (viitattu 03. 08. 2019).

[20] Axelson, J. *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*. 2. painos. Lakeview Research, 2007, 379 p.

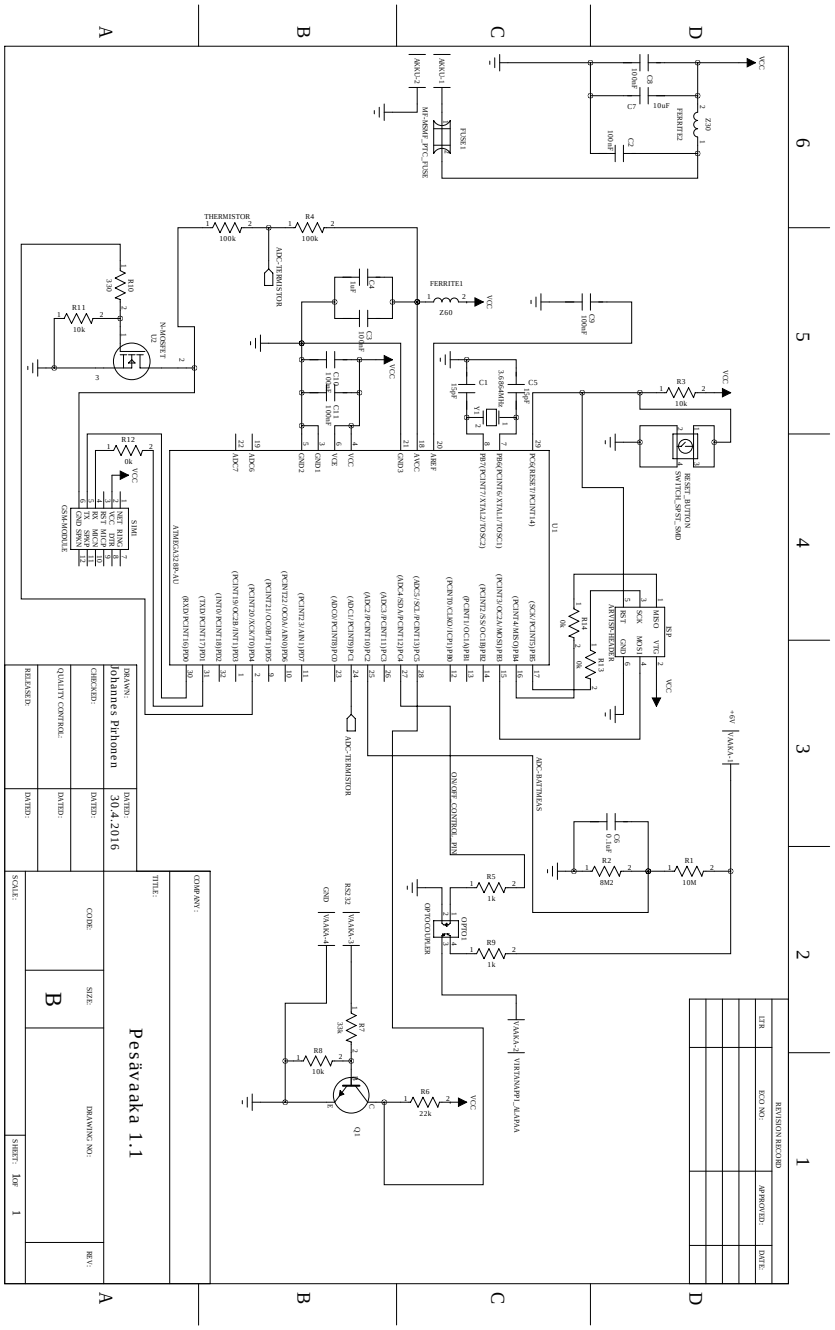
[21] *BC546B NPN Amplifier Transistor Datasheet*. BC546/D. ON Semiconductor. 2019, 6 p. URL: https://www.onsemi.com/pub/Collateral/BC546-D.PDF (viitattu 03. 08. 2019).

[22] Streetman, B. ja Banerjee, S. *Solid State Electronic Devices, Global Edition*. 7. painos. Pearson Education Limited, 2015, 621 p.

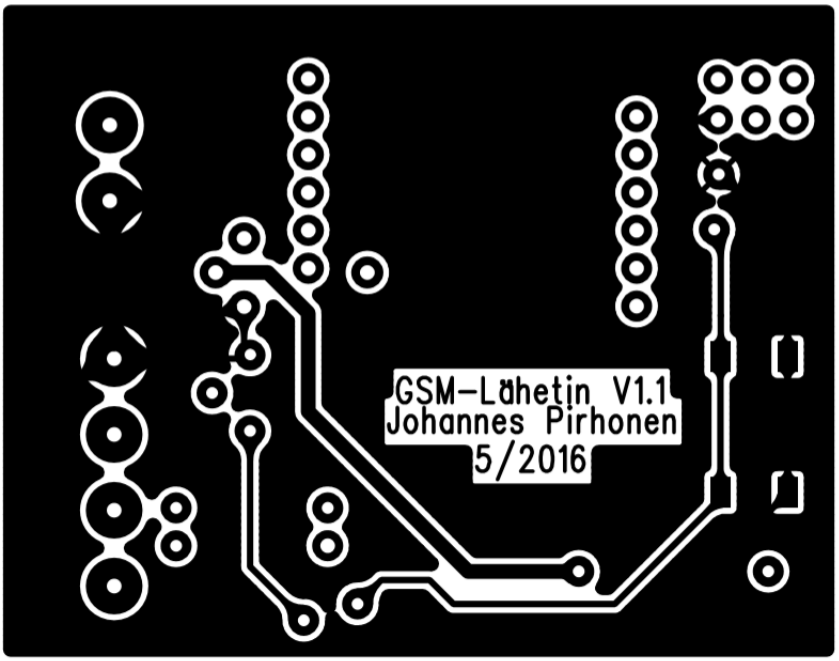
[23] Margolis, M. *Arduino Cookbook*. O'Reilly Media, Inc., 2011, 622 p.

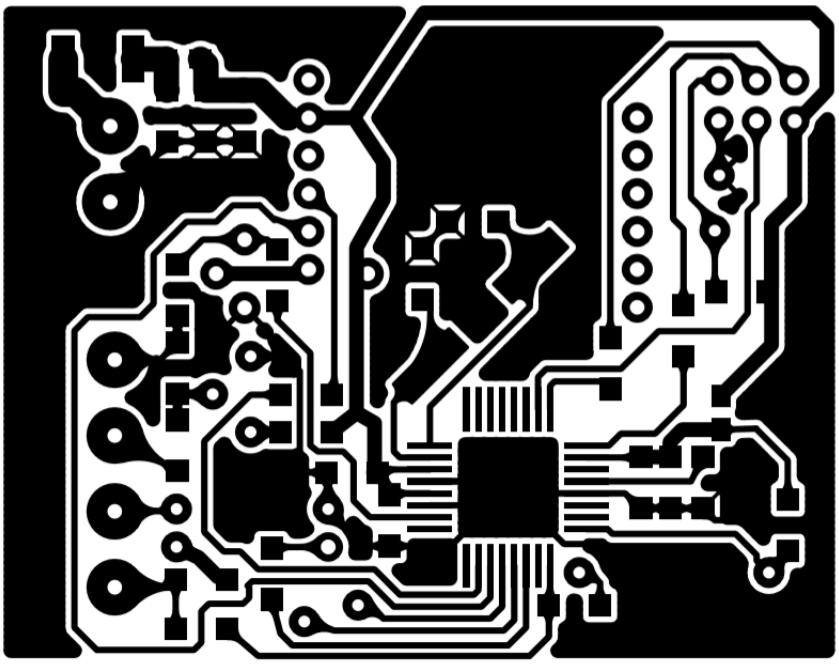
[24] *SIM800 Series AT-Command Manual V1.09*. Shanghai SIMCom wireless solutions Ltd. 3. elokuuta 2015, 380 p. URL: https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf (viitattu 23. 08. 2019).

A LAITTEEN PIIRIKAAVIO



B PIIRILEVYN VALOTUSMASKIT





C PÄÄOHJELMAN LOHKOKAAVIO